UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/032,567	01/02/2002	Jong-Deok Choi	YOR920010366US2	5834

48150 7590 07/17/2006

MCGINN INTELLECTUAL PROPERTY LAW GROUP, PLLC
8321 OLD COURTHOUSE ROAD
SUITE 200
VIENNA, VA 22182-3817

EXAMINER	
KENDALL, CHUCK O	
ART UNIT	**PAPER NUMBER**
2192	

DATE MAILED: 07/17/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)
	10/032,567	CHOI ET AL.
	Examiner	Art Unit
	Chuck O. Kendall	2192

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 17 April 2006.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-23 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-23 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on 01/02/02 is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

Detailed Action

1. This action is in response to communication filed 04/17/06.

2. Claims 1 – 23 are pending in this application.

Claim Rejections - 35 USC § 112

3. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

4. Claims 21 – 22 is rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention or to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention.

5. The 35 U.S.C. 112 1st paragraph rejections of Claim 21 was previously applied in the Advisory action of 09/21/2005 and in the previous Non-final rejection of 01/17/06 and is still being maintained in the action.

In summary Applicant continuous to argue for a limitation not taught in the specification being recited in the claims, i.e.,"tagging a statement with a set of threads".

In Applicant's most recent response 01/17/06 on page 10, 1st paragraph Applicant asserts that the limitation is taught in pages 19 and 21 and discloses that the statement "MustThreadObj(p(S.i)) and MayThreadObj(p(S.i))," (emphasis added), reasonably conveys to one of ordinary skill in the art and provides an adequate description of his claimed limitation of, "*tagging a statement with a set of threads*". Examiner still maintains that would not reasonably interpret this acronym and/or expression/object, i.e. "MustThreadObj(p(S.i)) and MayThreadObj(p(S.i))", to provide an adequate description of how to make or use the claimed limitation of "tagging a statement with a set of threads". So unless Applicant's specification is able to adequately convey and "describe with sufficient particularity such that one skilled in the art would recognize that the applicant had possession of the claimed invention" (MPEP, 2162), Examiner will maintain that it is not adequately described herein.

6. Claim 22 recites, "comparing sets of locks held by threads" in line 3. In Applicant's most recent response 01/17/06 on page 10, 1st paragraph Applicant alleges that the term "lock" is clearly enabled and identified on page 10, line 13 of his specification. In Applicant's specification on page 10, line 13 what is recited is "The invention then represents synchronized blocks, as well as synchronized methods, as separate nodes in the MCG". Again, Examiner doesn't see how it is reasonably and adequately conveyed and described with sufficient particularity, regarding the limitation of "*comparing sets of locks* held by threads", (emphasis added) as disclosed in claim 22.

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. Claims 1 – 20 and 23 are rejected under 35 U.S.C. 103(a) as being unpatentable over Petersen et al. USPN 6,593,940 B1 (hereinafter "Petersen) in view of Flanagan et al. USPN 6,343,371B1.

Regarding claim 1, Petersen discloses a method detecting a datarace in a multithreaded application (3:24 – 28), said method comprising:

inputting a set of input information (6:21 – 24, see "input to the code is a user's source code");

processing the set of input information by comparing threads that may execute statements in a statement pair (5:9 – 16, for *compare* see determine on a thread with respect to a different thread, and for *pair* see 13:40 – 45, for "determining when two or more threads...", "previous access" and "current access"); and

outputting a statement conflict set that identifies the statement pairs having execution instances which definitely or potentially cause data races (13:45 – 48,

"providing the indication that a race defect has occurred "). Although, Petersen doesn't explicitly disclose detecting the datarace statically i.e. without executing the multithreaded application, Petersen does perform disclose that static detection would be relied upon for conservative assumptions as to the behavior of the program being analyzed as supposed to providing exact behavior of the program (4:5 – 10).

However, Flanagan discloses statically detecting *potential* race conditions in multithreaded programs (4:10 – 15), (emphasis added). Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine Petersen and Flanagan, because it would enable detecting conservative assumptions/potential data race conditions.

Regarding claim 2, the method of claim 1, wherein the processing comprises:

selectively evaluating the input information with an IsPotentialDR relation for detecting potential dataraces (Flanagan, 4:10 – 15); and

selectively evaluating the input information with an IsDefiniteDR relation for detecting definite dataraces (Petersen, 13:33 – 35, for detecting race conditions).

Regarding claim 3, the method of claim 2, wherein, for a given pair of reference expressions, the IsPotentialDR relation comprises:

determining whether the reference expressions might be executed by different threads (negation of DefSameThreadObj) (Petersen, 5: 28 – 33, see deadlock);

determining whether the reference expressions might access the same field of the same object (Petersen, 5:28 – 33, see same set of locks, as interpreted by Examiner); and

determining whether the reference expressions might not be mutually synchronized (negation of DefSync) (Petersen, 4:52 – 57, see synchronization race).

Regarding claim 4, Petersen discloses, all the claimed limitations as applied in claim 2 above. Petersen, doesn't expressly disclose determining whether the reference expressions cannot be executed by the same thread (negation of PossSameThreadObj), determining whether the reference expressions must access the same field of the same object, determining whether the reference expressions cannot be mutually synchronized (negation of PossSync) and determining whether the reference expressions must execute.

However, Flanagan in a very similar configuration and analogous art teaches during statically detecting of potential race conditions (see Title in Flanagan), performing reduction of occurrences of false reports of potential race conditions, by flagging conditions such as "an object data field condition that is only accessed by a single thread", *(same as determining whether the reference cannot be executed by the same thread)* "inferring which object data fields are not shared among parallel executing threads" *same as determining whether the reference expressions must access the same field object* (Flanagan, 11:55 – 12:5) and building synchronization graphs *same determining whether the expressions cannot be mutually synchronized*, also see

(Flanagan 15:65 – 67, for generation edges in the graph representing execution paths) with regards to *determining whether the reference expressions must execute.*

Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine, Petersen and Flanagan because it would reduce false report occurrences (Flanagan,11:50 – 55).

Regarding claim 5, the method of claim 1, wherein the set of input information comprises at least one multithreaded context graph (Petersen, 6:1 – 5, see monitor lock cycle graph, as interpreted by Examiner).

Regarding claim 6, the method of claim 5, wherein the at least one multithreaded context graphs comprises an interprocedural call graph having each of a plurality of synchronized blocks as a separate node (Petersen, FIG. 2, see 214, and all associated text, also see 12:47 – 52, for support for synchronized conditions).

Regarding claim 7, the method of claim 5, wherein the at least one multithreaded context graph comprises an interprocedural call graph having each of a plurality of synchronized methods as a separate node (Petersen, FIG. 2, 206, for *method* see "ROUTINE").

Regarding claim 8, the method of claim 1, further comprising performing dynamic datarace detection on the statement conflict set (Petersen, 4:5, see dynamic analysis).

Regarding claim 9, the method of claim 1, further comprising performing escape analysis to identify statements that can access memory locations accessible by more than one thread (Petersen, 6:20 – 25, shows detecting data races, which is described in 4:34 – 36, same as *escape analysis*).

Regarding claim 10, the method of claim 1, wherein the processing comprises:

computing a node conflict set (Petersen, 6:25 –27, see error list); and

computing the statement conflict set by determining pairs of conflicting statements in the node conflict set (Petersen, 6:25 –27, see error list and viewing of errors and other defects).

Regarding claim 11, the method of claim 10, wherein the node conflict set computing comprises:

initializing a synchronization object set for each of a plurality of multithreaded context graph node (Petersen,12:50 – 55, shows synchronization events, also see 1:38 – 41 where it discloses that " most threading implementations supply synchronization mechanisms").

Regarding claim 12, the method of claim 11, wherein the node conflict set computing further comprises:

identifying all reachable conflicting node pairs for each thread-root node (Petersen,12:63 – 67, shows reporting tool which describes accessed pairs of threads).

Regarding claims 13, the method of claim 12, wherein the node conflict set computing further comprises:

identifying all reachable conflicting node pairs for each distinct pair of thread-root nodes in the at least one multithreaded context graph (Petersen,12:63 – 13:7, shows reporting tool and graph also see FIG.12,702, 716 and 714, "CALL TREE DISPLAY ", which would imply a hierarchy/root node); and

identifying all reachable conflicting node pairs for each thread-root node in the at least one multithreaded context graph that is invokeable by more than one thread (Petersen,12:63 – 13:7, shows reporting tool and graph also see FIG.12,702, 716 and 714 "CALL TREE DISPLAY ", which would imply a hierarchy/root node).

Regarding claim 14, the method of claim 1, wherein the input comprises meta-information relating to said multithreaded application which is written in an object-oriented programming language (Petersen, 3:36 – 38, see "Java").

Regarding claim 15, the method of claim 1, wherein the input comprises a multithreaded context graph for said multithreaded application written in an object oriented programming language (Petersen, 3:36 – 38, see "Java", also see FIG. 12, which shows class and jar files, which also would indicate further the use of the Java

object oriented language).

Regarding claim 16, the method of claim 15, wherein the input further comprises a plurality of bytecodes that collectively comprise the application (Petersen, 3:36 – 38, see "Java", bytecodes are inherent in Java).

Regarding claim 17, Petersen discloses a method detecting a datarace in a multithreaded application (3:24 – 28), said method comprising:

an input interface (10:33 – 35, see graphical user interface 700); an output interface (10:30 – 35, see display window);

a storage medium comprising the application and meta-information relating to the application (11:45 – 48, see gathers and sorts information, also see 14:20 – 25, for storage medium); and determine a statement conflict set (SCS) for the application (6:25 –27, see error list and viewing of errors and other defects). Petersen doesn't explicitly disclose processing the application and the meta-information without executing the application.

Although, Petersen doesn't explicitly disclose processing the application and the meta-information (detecting datarace) without executing the application, Petersen does perform disclose that static detection would be relied upon for conservative assumptions as to the behavior of the program being analyzed as supposed to providing exact behavior of the program (4:5 – 10).

However, Flanagan discloses statically detecting <u>*potential*</u> race conditions in multithreaded programs (4:10 – 15), (emphasis added). Therefore it would have been

obvious to one of ordinary skill in the art at the time the invention was made to combine Petersen and Flanagan, because it would enable detecting conservative assumptions/potential data race conditions.

Regarding claim 18, the computer processing system of claim 17, wherein the meta-information comprises a multithreaded context graph (Petersen, 6:1 – 5, see monitor lock cycle graph, as interpreted by Examiner).

Regarding claim 19, the computer processing system of claim 17, wherein the processor is further configured to perform dynamic datarace detection on the statement conflict set (Petersen, 4:5, see dynamic analysis).

Regarding claim 20, the computer readable program product version of claim 17, see rationale above as previously addressed and regarding computer readable program product see (Petersen, 14:20 – 60).

Regarding claim 21, the method of claim 1, wherein said comparing said threads comprises:

tagging a statement with a set of threads that may execute said statement and comparing sets of threads for said statements (Petersen, 13:35 – 45).

Regarding claim 22, comparing sets of locks held by threads that may execute said statements (Petersen, 13:35 – 45).

Regarding claim 23, Petersen discloses a method detecting a datarace in a multithreaded application (3:24 – 28), said method comprising:

inputting a set of input information (Petersen, 6:21 – 24, see “input to the code is a user’s source code”);

processing the set of input information by comparing threads that may execute statements in a statement pair (5:9 – 16, for *compare* see determine on a thread with respect to a different thread, and for *pair* see 13:40 – 45, for “determining when two or more threads…”, “previous access” and “current access”); and

outputting a statement conflict set that identifies the statement pairs having execution instances which definitely or potentially cause dataraces (Petersen,13:45 – 48, “providing the indication that a race defect has occurred ”);

performing dynamic datarace detection (Petersen, 4:5, see dynamic analysis), on the Statement Conflict Set and computing the Statement Conflict Set by determining pairs of conflicting statements in the node conflict set (Petersen, 6:25 –27, see error list and viewing of errors and other defects), wherein said computing the node conflict set comprises:

initializing a synchronization object set for each node in multithreaded context graph (MCG) (Petersen,12:50 – 55, shows synchronization events, also 1:38 – 41 “most threading implementations supply synchronization mechanisms);

identifying all reachable conflicting node pairs for each thread root node (Petersen,12:63 – 67, shows reporting tool which describes accessed pairs of threads);

identifying all reachable conflicting node pairs for each distinct pair of thread-root nodes in the multithreaded context graphs (Petersen,12:63 – 13:7, shows reporting tool and graph also see FIG.12,702, 716 and 714, "CALL TREE DISPLAY ", which would imply a hierarchy/root node); and

identifying all reachable conflicting node pairs for each thread-root node in the multithreaded context graphs that is invokeable by more than one thread (Petersen,12:63 – 13:7, shows reporting tool and graph also see FIG.12,702, 716 and 714 "CALL TREE DISPLAY ", which would imply a hierarchy/root node), and

wherein the input comprises said multithreaded context graph (MCG) for a multithreaded application which is written in an object-oriented programming language (Petersen, 3:36 – 38, see "Java").

Although, Petersen doesn't explicitly disclose processing the application and the meta-information (detecting datarace) without executing the application, Petersen does perform disclose that static detection would be relied upon for conservative assumptions as to the behavior of the program being analyzed as supposed to providing exact behavior of the program (4:5 – 10).

However, Flanagan discloses statically detecting *potential* race conditions in multithreaded programs (4:10 – 15), (emphasis added). Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine

Petersen and Flanagan, because it would enable detecting conservative assumptions/potential data race conditions.

Regarding claim 24, wherein said performing said dynamic datarace detection is performed after said outputting said statement conflict set (Petersen, 4:5, see dynamic analysis).

Regarding claim 25, the method of claim 1, wherein said outputting said statement conflict set comprises outputting said statement conflict set to at least one of a display device, a printer a communication interface and a storage media interface (FIG.7, 718).

Response to Arguments

11. Applicant's arguments with respect to claims 1 – 23 have been considered but are moot in view of the new ground(s) of rejection.

Correspondence information

12. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chuck Kendall whose telephone number is 571-272-3698. The examiner can normally be reached on 10:00 am - 6:30pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is **571-273-8300**.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Ck.

Chuck Kendall 7/26/05